**Full Stack Development with MERN**

**1. Introduction**

**Project Title**: Book a Doctor using MERN  
**Team Members**:

* Shai Subramaniam G J – Front-end Developer
* Kevin Joseph J – Full-stack Developer
* Serabin Rojer A – Back-end Developer
* Keerthivasan V – Technical Lead

**2. Project Overview**

**Purpose**:  
This project aims to create an efficient and user-friendly online doctor booking platform. Patients can search for doctors, book appointments, and manage their medical history. Doctors can manage their availability and appointments while admins oversee platform activities.

**Features**:

* User registration and login with JWT authentication.
* Doctor profile creation and appointment scheduling.
* Patient dashboard to view bookings and manage medical records.
* Admin dashboard to manage users, doctors, and appointments.
* Real-time notifications for appointment updates.

**3. Architecture**

**Frontend**:

* Built using React, the frontend provides a seamless user experience.
* Includes features such as user authentication, doctor search, appointment booking, and notifications using React Router for smooth navigation.

**Backend**:

* Built using Node.js and Express.js, the backend manages API requests, user authentication, and database operations.
* Controllers handle user and doctor data, along with appointment management.

**Database**:

* MongoDB stores data on users, doctors, appointments, and medical records.
* Collections are organized for efficient querying and data updates.

**4. Setup Instructions**

**Prerequisites**:

* Node.js v14+
* MongoDB v4+
* (Optional) npm or yarn for package management

**Installation**:

1. Clone the repository: git clone <repository-url>
2. Navigate to both the backend and frontend directories and install dependencies:

For Backend: cd backend

npm install

For Frontend: cd ../frontend

npm install

1. Set up environment variables by creating a .env file in the backend directory with database connection strings, JWT secret, etc.

**5. Folder Structure**

**Client**:  
The frontend directory contains:

* src/components - Reusable React components (e.g., DoctorCard, AppointmentForm).
* src/pages - Pages like Home, Login, DoctorProfile, and BookingDashboard.
* src/utils - API calls and helper functions.
* src/styles - CSS files for styling.

**Server**:  
The backend directory is organized as follows:

* config - Database connection configuration.
* controllers - Functions to manage appointments, users, and doctors.
* routers - API routes for handling requests.
* middlewares - Authentication and error-handling middleware.
* schemas - MongoDB schemas/models for users, doctors, and appointments.

**6. Running the Application**

**Frontend:**  
Start the frontend server: npm start

**Backend:**  
Start the backend server: npm start

**7. API Documentation**

The backend exposes several endpoints, including:

* **POST** /api/auth/login - User login with JWT authentication.
* **POST** /api/auth/register - User registration.
* **GET** /api/doctors - Fetch available doctors.
* **POST** /api/appointments - Book an appointment.

**Example Response:**

{

"status": "success",

"data": {

"doctors": [

{

"id": "doctor-id",

"name": "Dr. John Doe",

"specialization": "Cardiology",

"availability": ["2024-11-16T10:00", "2024-11-16T11:00"]

}

]

}

}

**8. Authentication**

Authentication is handled using JWT tokens. After login, users receive a token stored in local storage. Protected routes (e.g., Booking Dashboard) require a valid token, verified using middleware.

**9. User Interface**

The user interface includes:

* A login and registration form.
* Doctor search page with filtering by specialization and location.
* Booking dashboard for patients to manage appointments.
* Profile page for doctors to update availability and view appointments.

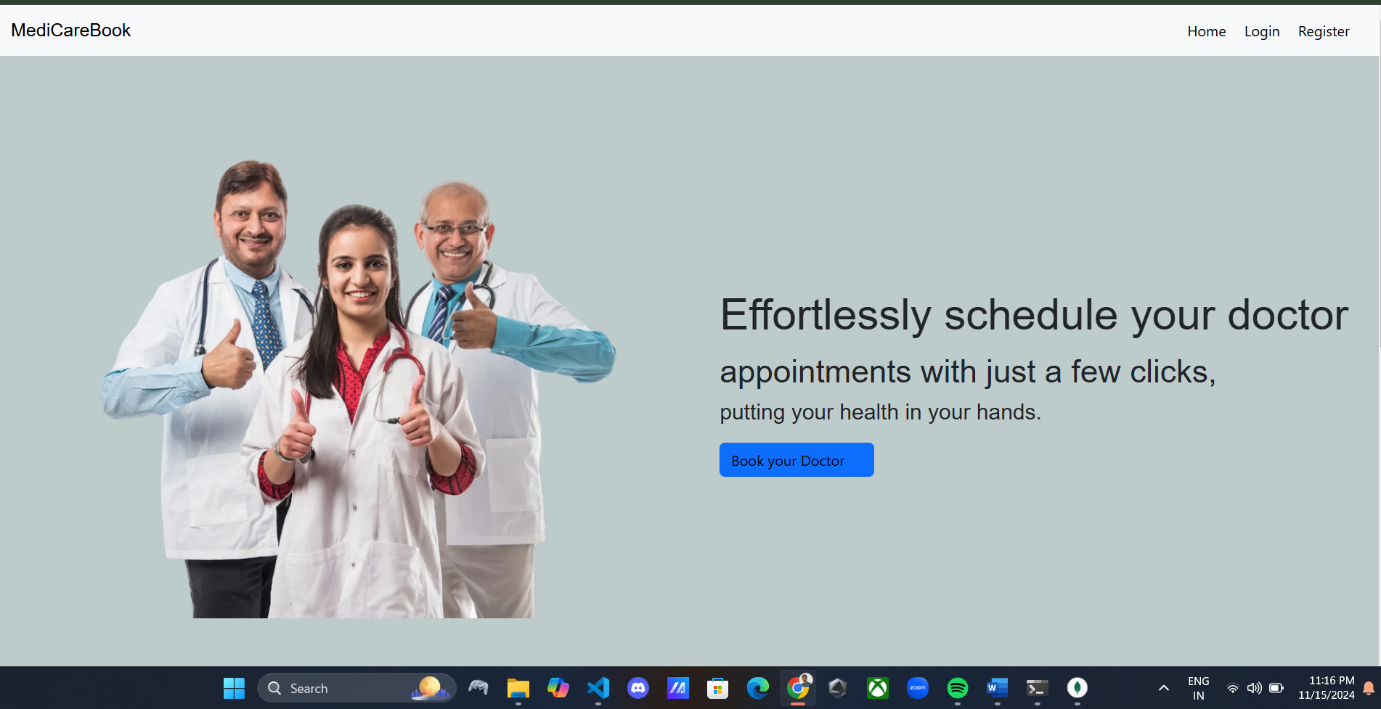
#### **10. Testing**

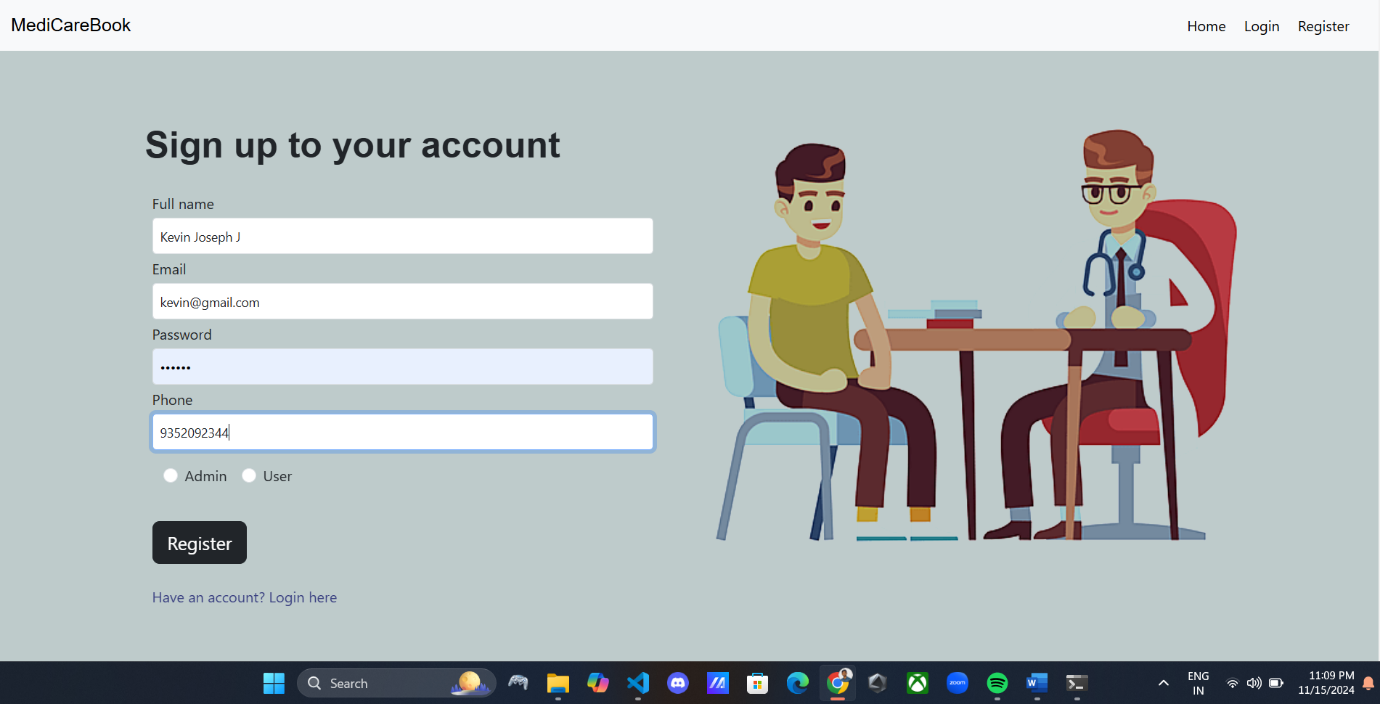
Testing is performed using Jest and Postman.

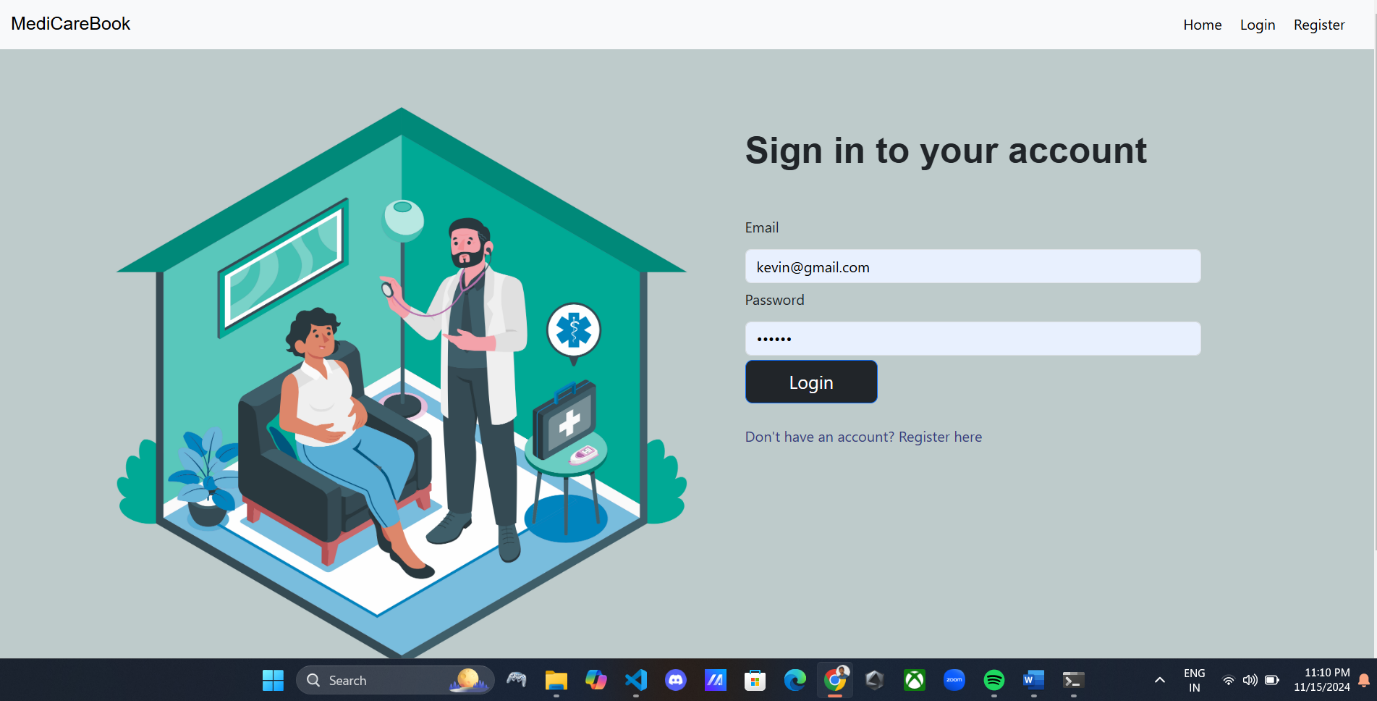
* **Unit Tests**: Validate React components and backend API functionality.
* **Integration Tests**: Ensure seamless interactions between frontend and backend.

**11. Screenshots**

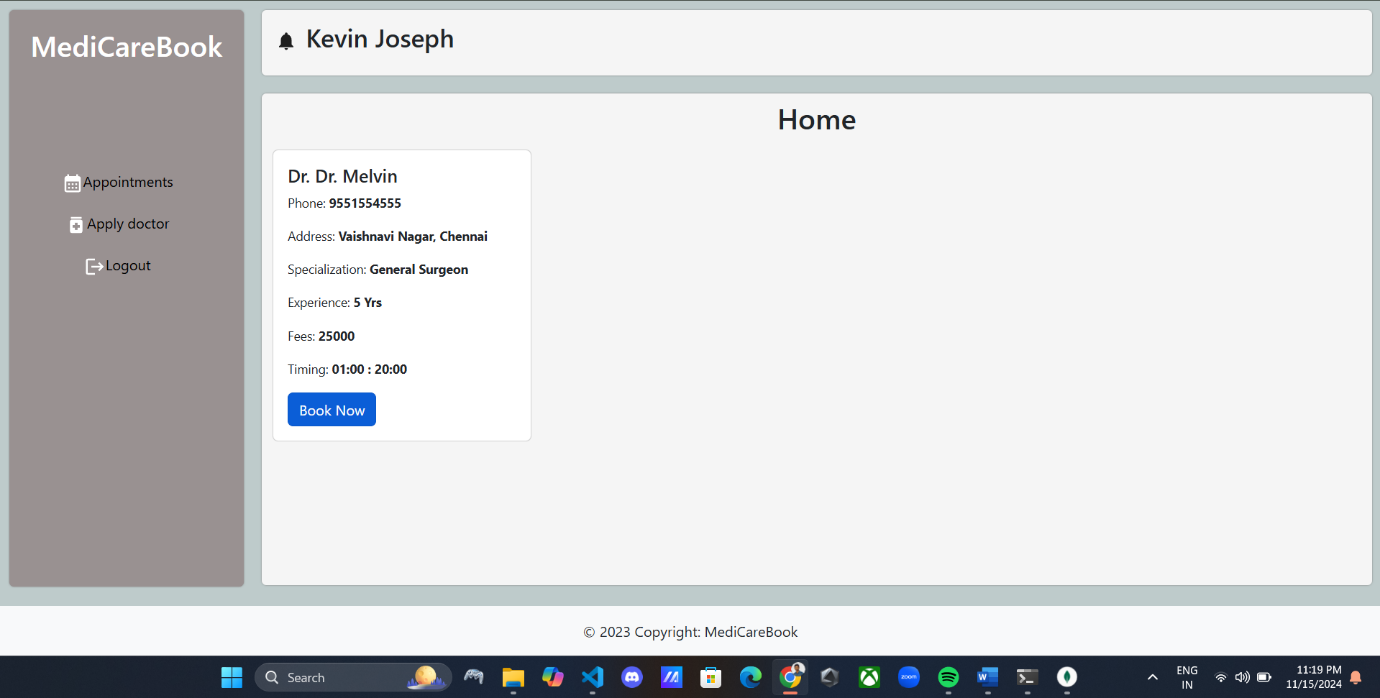
* A login and registration form.

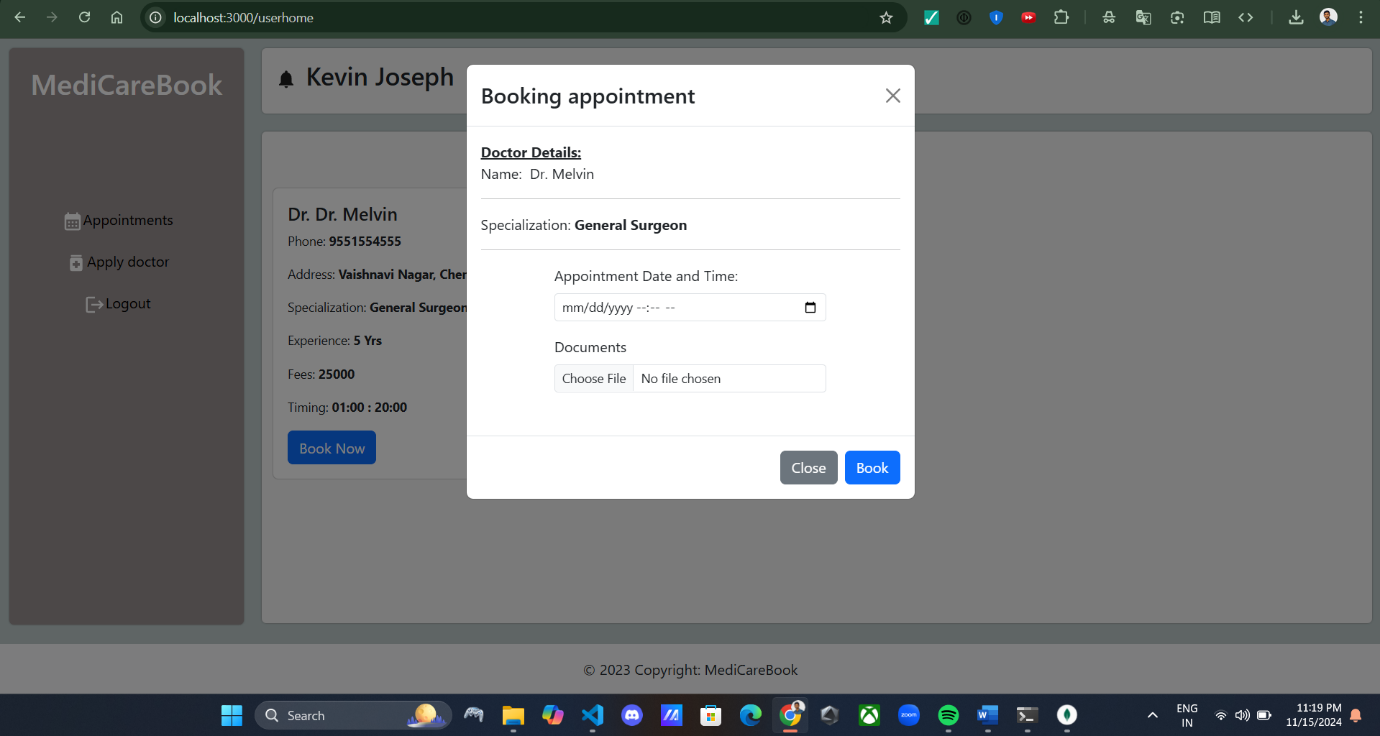




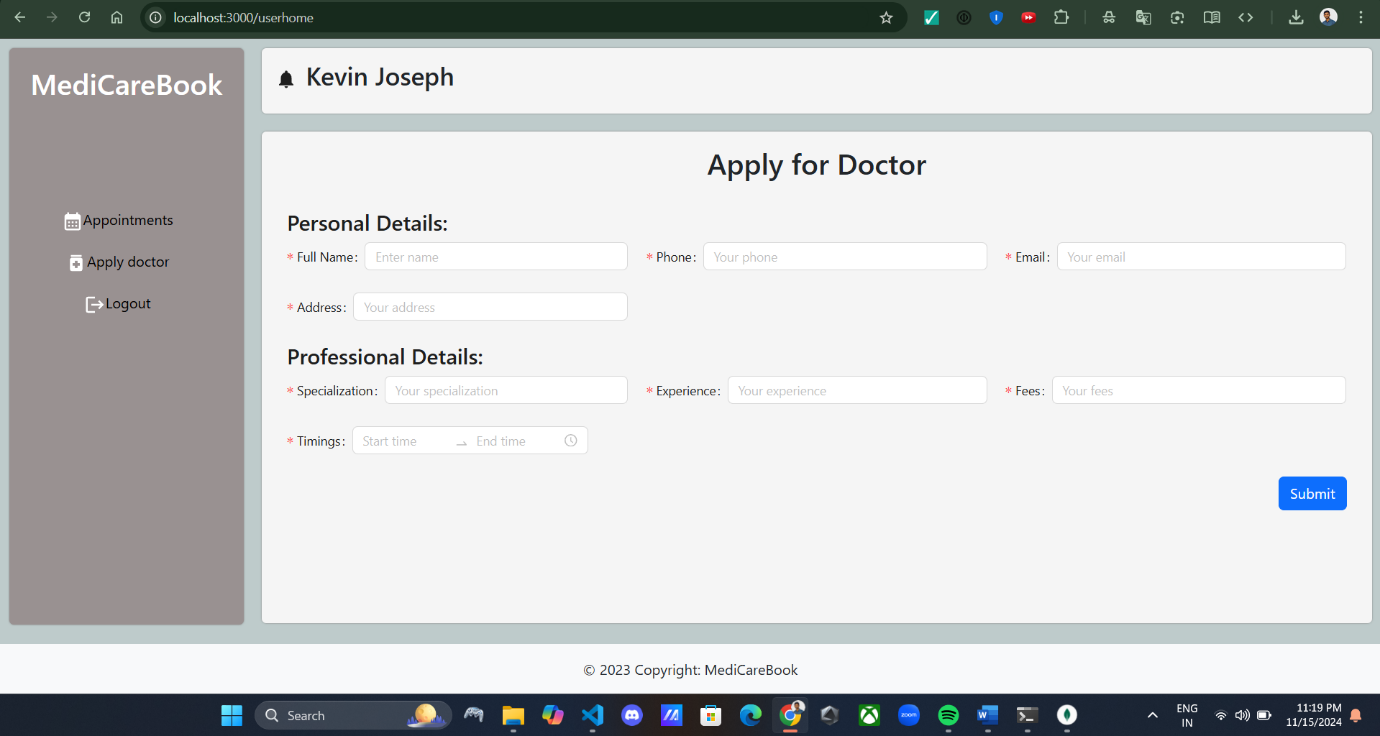


* Booking a Doctor appointment

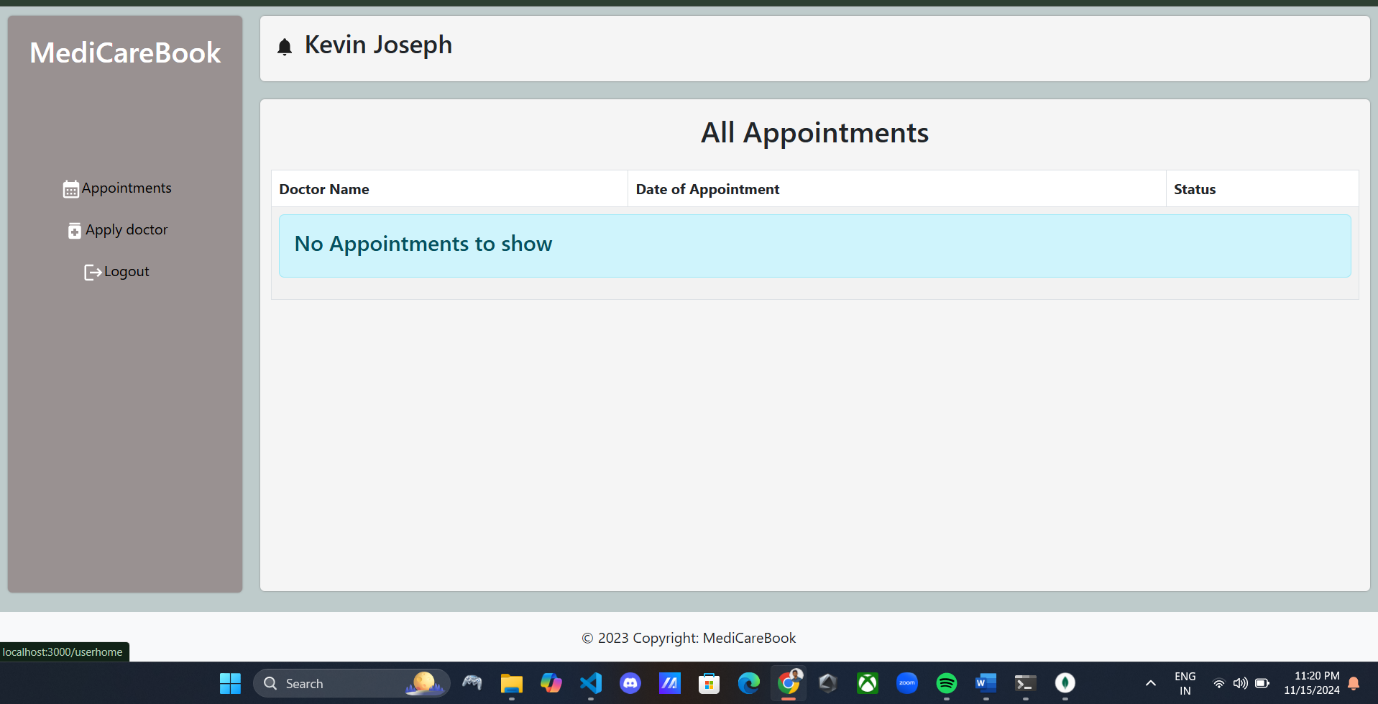


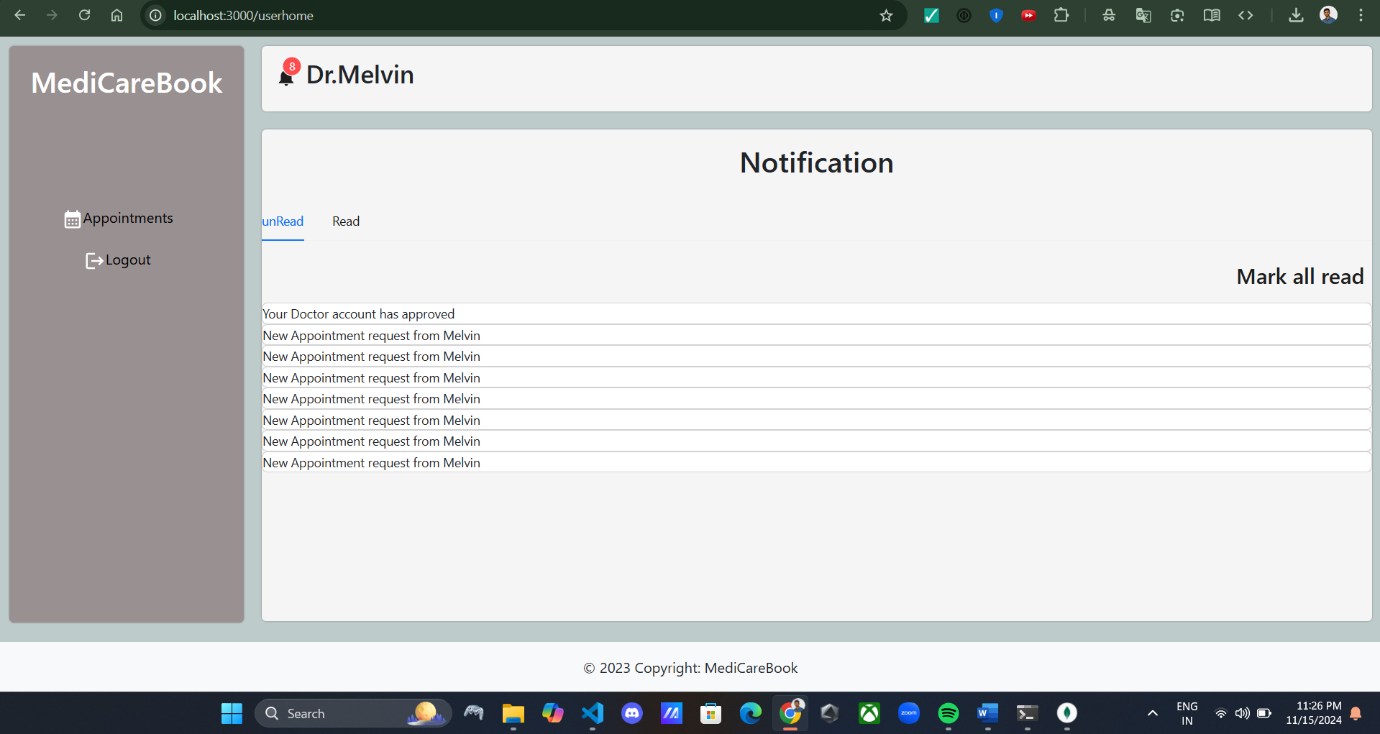


* Application for a Doctor



* Appointments page for the patients



* Admin Page for Doctors

.

**12. Known Issues**

* **Token expiration**: Users may be logged out unexpectedly when their JWT token expires. Implementing a token refresh mechanism can resolve this issue.
* **Real-time messaging delays**: Occasional delays in sending or receiving notifications or messages during appointment scheduling. Optimizing the socket connection can help.

**13. Future Enhancements**

* **Patient medical history upload**: Allow users to upload past medical records for doctors to review.
* **Doctor availability status**: Implement real-time availability tracking for doctors.
* **Payment gateway integration**: Enable patients to pay for consultations securely using payment gateways.
* **Video conferencing**: Add support for telemedicine consultations with video calling between doctors and patients.
* **Enhanced analytics**: Provide doctors and administrators with detailed reports and analytics on appointments and user engagement.